**0616018 林哲宇, 0616032 張哲銓**

1. **算 cycles**
2. **Simulated program execution cycles**

計算組合語言中指令的總和，也就是最一開始初始化的兩個addi，接著分別是loop\_i, loop\_j, loop\_k的指令數，其中loop\_k內的指令會被執行 m\*n\*p次，但是在最後跳出迴圈的slt和beq只會被執行m\*n次；同樣的，loop\_j內的指令會被執行m\*n次，但是最後跳出迴圈的slt和beq只會被執行m次；最後loop\_i內的指令只會被執行m次，並且加上最後跳出迴圈的slt和beq。全部加起來後再加上exit。

1. **One-word-wide memory organization**

因為block size是8 words，一次只能send 1 word，所以miss penalty為：

1 + 8 x (1 + 2 + 100 + 1) + 2 + 1 = 836

hit時需要的cycle為：

1 + 2 + 1 = 4

1. **Wider memory organization**

因為可以一次send 8 words，所以miss penalty 為：

1 + (100 + 1 + 2 + 1) + 2 + 1 = 108

hit 時需要的cycle 為：

1 + 2 + 1 = 4

1. **Two-level memory organization**

有兩個level，當L1 miss時的miss penalty：

1 + 4 x (1 + 10 + 1 + 1) + 1 + 1 = 55

L2 miss時的miss penalty：

1 + 32 × (1 + 100 + 1 + 10) + 4 × (1 + 10 + 1 + 1) + 1 + 1 = 3639

Hit時需要的cycle為：

1 + 1 + 1 = 3

1. **三種memory organizations差異**

Wider memory organization比One-word-wide memory organization的memory stall cycles少的原因很單純是因為前者的一次能send 8 words。但是Two-level memory organization在a1xb1和a2xb2中的stall cycles比較多，卻在a3xb3和a4xb4比較少的原因是a1xb1和a2xb2的矩陣太小。由於在一開始cache中沒有任何值的時候一定會產生必須的miss，這讓Two-level memory organization的miss penalty在矩陣小的時候，stall cycle看起來比較大；但是當矩陣變大時，Two-level memory organization的優勢就顯現出來了，因為L2有較大的cache可以存放，所以大多數的miss都只需要從L1往L2找，不需要去memory，然而前面兩個仍然需要把資料從memory送回cache，因此Two-level memory organization在矩陣大時需要的cycle就比較小。

1. **Bonus**

**1. 作法**

將 “temp1 = 4(i\*p+j) + C[]base” 的五行指令和sw移出loop\_k外，移至loop\_j，並直接拿$10加上$19，結束loop\_k後才存回 temp1，取代原先$20做的事。這樣一來必定就可以減少execution cycles。

**2. 結果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| case | Execution cycles | One-word-wide memory organization | Wider memory organization | Two-level memory organization |
| a1xb1 | 1265 | 5632 | 1264 | 11856 |
| a2xb2 | 4989 | 17536 | 4432 | 31584 |
| a3xb3 | 137845 | 7565120 | 1006568 | 772096 |
| a4xb4 | 4221381 | 251049600 | 33230544 | 1002432360 |